# A crowdsensing platform for mass surveillance

Attila Mátyás Nagy [1], Vilmos Simon [2]
[1] Budapest University of Technology and Economics, 1111, Hungary
[2] Budapest University of Technology and Economics, 1111, Hungary
*anagy@hit.bme.hu*

***Abstract*— Nowadays festivals and city events attract huge number of visitors and unfortunately in large masses a minor panic could have incalculable consequences despite of the organizers' efforts to do everything for participants' safety. Therefore we have developed an integrated mass surveillance system based on crowdsensing data, where the system helps authorities and organizers in information gathering about the actual dynamics of the crowd. Based on real time and representative information, they are able to perform fast and targeted interventions.**

***Keywords*— *mass surveillance; mobile crowd sensing; smart cities***

## I. INTRODUCTION

How can we prevent the emergence of a mass panic? How can we detect that the crowd reached the critical density which could cause a mass incident? How can we forebode the formation of an emergency situation? Unfortunately, there are crowd disasters each year and some of them could be avoided if the organizers have had more information about the crowd. Perhaps the most notorious disasters were in Mecca during the Hajj. Last year more than 1100 people died and 900 injured when a very large crowd of people squeezed and trampled each other [1].

As the number of different types of mass events increases year by year, crowd incidents would become a more common misfortune, if we don not do anything to prevent them. The authorities need reliable surveillance systems which can provide better monitoring solutions to understand the movement of the crowd in a real time manner. If the organizers and authorities can obtain valid information about the crowd dynamics, they would be able to perform quick and targeted interventions.

The vast majority of visitors of today's large events already have a mobile device (such as smart phones and tablets) which has various built-in sensors (GPS, accelerometer, gyroscope). Data from these sensors are collectable and by processing them valuable information can be obtained, thus it is possible to monitor crowd dynamics, and based on the history of the given venue to estimate future states. This is a typical case of crowd sourcing. Crowd sourcing is a way of cooperation where large group of users are solving different simple tasks (which are part of a bigger task), so complex problems can be handled more efficiently. Crowd sensing [2][3] is a subtype of crowd sourcing where the outsourced job is a sensing task. This can be very useful for several reasons. As organizers of large music festivals have explained to us, for such large events, although there are models and inaccurate estimates of distribution of the crowd, we can not exactly know how many people are staying in one area or moving to another one at a given moment. If we could measure it via a mobile crowd sensing platform, depending on the distribution of the crowd we could reallocate the security and medical personnel more precisely, furthermore we could also dynamically modify the evacuation plans if needed. The mobile crowdsensing platform should also allow us to send different messages to participants based on their position. These messages may include public notices, and even promotional messages.

This kind of a system needs to be prepared for a large amount of incoming data. On the one hand it also means that the system should be able to handle a large number of simultaneous connections, tens of thousands simultaneous connections would send data at the same time. On the other hand, it should be able to efficiently process large amount of measurements on a distributed server cluster, whereas the results are needed in real time.

## II. RELATED WORKS

There are different systems that perform various surveillance tasks, however they are not necessarily addressing the tasks the organizers are so keen to solve nowadays.

Camera-based surveillance systems (CCTV) are the most common, only in the United Kingdom millions of surveillance cameras has been installed [4]. They are applied for several purposes, of course, not only to monitor large events[5][6]. The camera-based systems' major shortcoming is that they are not able to give a comprehensive picture of the current status of the crowd. When placing lot of cameras at an event, the communication overhead of collecting the stream of hundreds of cameras would be enormous. Furthermore aggregating those streams and making precise calculation about the mass size would be almost impossible. However, these systems can effectively work together with a crowd sensing system, especially if we mount cameras on UAVs[7], being able to pilot them at area of the event, adapting their trajectory to the crowd distribution, obtained by a mobile crowdsensing system.

A Bluetooth-based solution [8] is utilized as well, here the goal was to create a mobile application which can assist in the management and in the communication tasks of the organizers for large-scale outdoor events. To determine the size and the distribution of the crowd it uses the Traffax sensors[9], which are placed near the entrances of the event so they can count the active Bluetooth devices passing by. This approach has limited capabilities, as it can only give us the density of the participants near the gate sensors, but has no knowledge about the internal state of the crowd. Of course, it is possible to place

more gates at the area of the event, but it can be extremely costly in those cases.

Another system [10] uses GPS positioning and WiFi/ GSM-fingerprinting [11] methods to determine the position of pedestrians. A festival application have been developed which gathers position information from participants at every second. These location data samples are transmitted to the mobile server where they are evaluated. After the evaluation, a heat map is generated to display the actual state of the crowd. The system assumes that the visitors using the application are statistically distributed in proportion to the crowd so correct conclusions can be made about the global behavior of the crowd. Naturally, if there are more active users in the system the heat map becomes more accurate, but the integration of a camera-based system can also improve the accuracy. The system have been tested at the Lord's Mayor Show 2011 in London and at the Vienna City Marathon 2012. A major flaw in the system is that the mobile application queries the GPS position every second. This approach depletes the device's battery really quickly therefore the participants will not use it in a long term.

The previously introduced systems can help the work of the organizers and authorities in different ways, but there is a clear need from the organizers for an integrated solution which combines the benefits of the previous systems and provides more accurate information for the organizers while it generates less network traffic and consumes much less energy at the devices.

## III. Development of a Mass Surveillance System

### A. System requirements

As we mentioned in the introduction, our new system should be feeded by the mobile crowd's sensing data. It means that information is received from individuals of the crowd. To make that possible, a mobile application is needed which monitors the visitors movement and position. When monitoring a large crowd our system needs to be prepared to receive data from more than ten or hundred thousands of different sources. Thus the system should be able to maintain a huge number of parallel communication channels. On the other hand, the large number of connections will generate remarkable network traffic so it is crucial for the system to efficiently process a large amount of measurements, by distributing them on server clusters, to enable the authorities to follow the crowd's movement tendencies in real time.

Taking into consideration the previous findings, if we collect data from each user continuously it will cause an unmanageable set of data and it will also create lot of unnecessary states to be stored in the system, depleting the device batteries pretty quickly. It seems a better idea if we collect less data, but they contain relevant information.

### B. Mass surveillance model

As a first step, we have studied thoroughly the literature of crowd dynamics models, to address the problems mentioned above. The known models are always based on a physical model, since the movement of human masses shows a strong resemblance to dynamics of liquids and granular materials[12][13]. We have also studied discrete (cell based) mass surveillance models [14][15][16][17][18] and we found that although all of these approaches are working properly in simulation scenarios, we had to take into account other considerations which are really important in real life applications. Thus we have modified the cellular model to provide an adequate abstraction which allows us to reduce significantly the number of irrelevant data and the network traffic.

First, we divided the space to discrete areas, so-called cells. We have to highlight that the cells are not squares like in other models[14][15], but they are arbitrary convex quadrangles. It is better to define areas which need to be examined as one coherent unit in one cell instead of dividing them to two or more cells. Convex quadrangles support this approach very well. For example, it is not appropriate when a cell is defined in a way that part of its area is covering a busy road, while the other part covers a building which will obviously block the movement of pedestrians inside the cell, actually breaking it

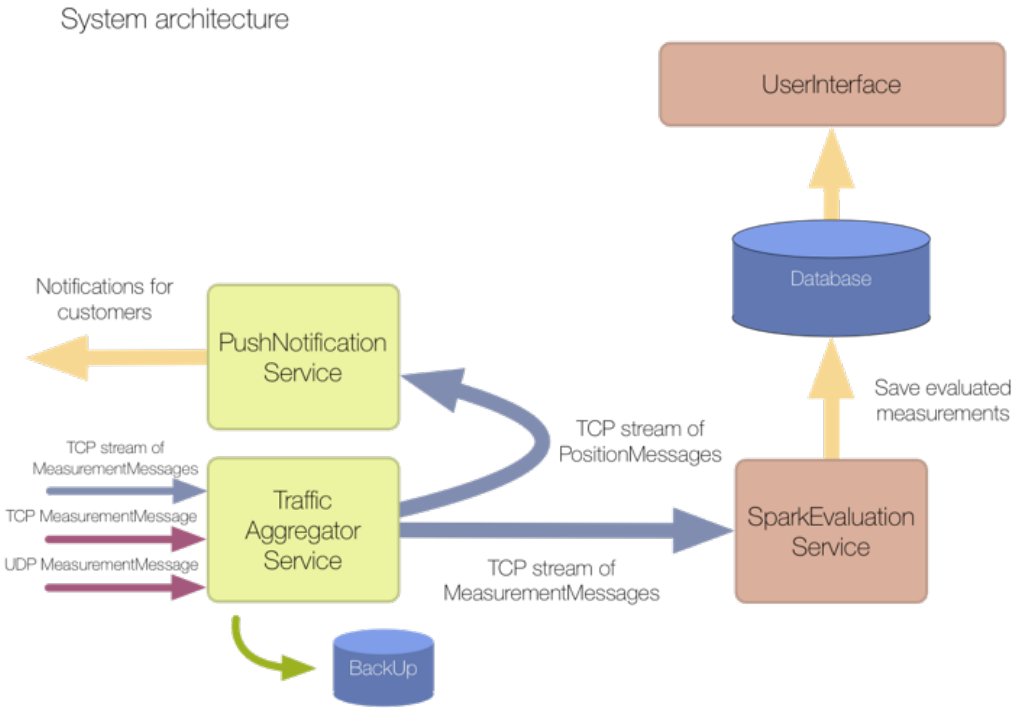


Figure 1. System architecture

down to two separate cells. We have to know each cell size for density calculation but they are calculated easily based on GPS coordinates.

Secondly, we had to place the participants in our model. We assume that a participant's mobile device has a built-in GPS sensor, a gyroscope and accelerometer. The data from these sensors is used to determine the participant's position and velocity which are required to make correct evaluations about the crowd dynamics. By collecting samples from the gyroscope and accelerometer, the GPS needs to be activated only from time to time to obtain a more precise position, in the meanwhile the position of the user can be predicted in an energy efficient way, by utilizing the dead reckoning [26][27] method. This can be precise enough in the majority of the cases, as only the cell should be identified where the user resides, not the accurate geographical position (due to the cell based model). This means that the GPS sensor does not need to be active all the time of the surveillance (like in the above described system, where the samples were extracted every second), therefore the batteries of the participants' smartphones will not be depleted in short time.

Therefore it is unnecessary to send the exact GPS coordinates, it is enough to send only the previous and the next cells' identifier. It speeds up the calculation because of two reasons. First is that we can aggregate measurement messages to cells so we do not have to use GPS coordinates during the evaluation phase. Second is that the devices send their measurement messages only when changing a cell, as in a real life scenario a participant is often standing still (watching a concert or speaking with other persons), not changing a cell for dozen of minutes (or even hours), therefore the communication and computation overhead can be decreased significantly.

As already mentioned, another advantage is that the mobile application has low energy consumption as it uses dead reckoning instead of using periodically GPS position queries when it tries to figure out where is it.

In our system the current status of a participant (or device, because they are the same in this case) can be described by two variables: the position (cell id) and the velocity of the person.

Based on this two metrics we calculate the following crowd characteristics for each cell:

- crowd density
- crowd velocity variance
- crowd pressure[19]

Crowd density is a very useful metric (it can be calculated by dividing the headcount of a cell with the area of a cell), as it can be a warning sign of critical crowd conditions [19]. With velocity variance we can predict unexpected incidents (for example a stampede), while crowd pressure is the decisive variable quantifying the hazard to the crowd (it can be calculated by multiplying the crowd density with the variance of velocity).

### *C. System architecture*

In our integrated mass surveillance system a number of separate components work together (Figure 1). The different tasks have been distributed among these units so a component failure does not cause the failure of the whole system.

As we mentioned in the Requirements section, our system has to be able to receive incoming data (measurement messages) from more than ten or hundred thousands of different sources (from participants' devices). Thus our first job was to create a component (TrafficAggregatorService) which is able to handle this enormous amount of sources and aggregate their data to streams. Other important tasks to be solved were the evaluation of measurement messages and the position based notification mechanism for participants, so we have developed appropriate system components for them also. The unit responsible for the evaluation of measurement messages became the SparkEvaluationService, which is a cluster computing unit, based on Apache SparkStreaming[25]. It simply connects to the TrafficAggregatorService, downloads the continuous stream of data, runs our evaluation algorithms

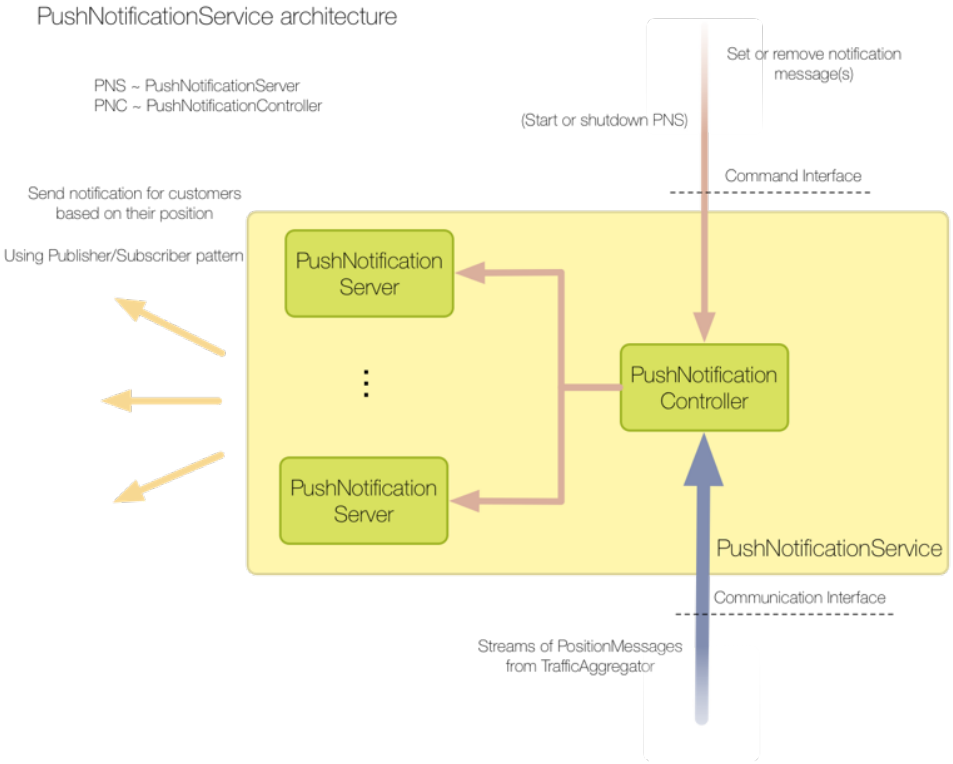


Figure 2. PushNotificationService architecture

on them, and finally saves the results to the database.

The PushNotificationService component is an own implementation of a general push notification service. We developed it because we did not want to depend on another third party service, this way we could customize and optimize it for our purposes. As the SparkEvaluationService, this component also connects to the TrafficAggregatorService, but it downloads only the position information and based on these it sends notification messages for the participants, when needed.

It is not shown on Figure 1. , the participants' devices keep connection via a mobile application. As majority of the large events have their own applications, we developed a library with an API which can be simply attached to an existing application. In the background it collects movement information from sensors and sends its measurement messages to the TrafficAggregatorService. Via the API of the library, a participant can subscribe to get various notification messages from the PushNotificationService, however in emergency situation, the warning message can be sent out in this interface also.

*1) TrafficAggregatorService*

As we can see on Figure 1, the TrafficAggregatorService (TAS) provides an interface for mobile applications to forward their measurement messages which contain the position (cell id) and velocity information. From this interface all received data is forwarded into TCP streams for the PushNotificationService and for the SparkEvaluationService. Naturally, TAS can provide more streams for the evaluation service if multiple cluster nodes are in operation at the SparkEvaluationService, furthermore it can also provide more streams for the PushNotificationService if necessary.

To be able to cope with the heavy loads of measurement messages, the TAS uses Netty [22], an asynchronous event-driven framework at the mobile application interface. With Netty, TAS can handle tens of thousands simultaneous connections. For more efficient resource utilization there is no continuous connection between the applications and the TAS, the devices send the measurement message (a message is compressed in 29 bytes) only when they change the cell (as the mobile application downloads at the beginning the cell structure of the event).

As the sent data packets are pretty small, in most cases the packets of the TCP three-way handshake would be an unnecessary communication overhead compared to them, thus TAS provides TCP and UDP interfaces too for efficient communication. In general the mobile applications send data using the UDP interface, but in some emergency situation data are sent via the TCP connection to ensure measurements are successfully received by the TAS.

Sometimes one running TAS instance is not enough to serve huge number of incoming measurement messages. In this case, it is possible to chain more TAS instances together in a master-slave architecture, so the load capacity will increase. In this case the PushNotificationService and the SparkEvaluationService connect to the master instance which receives all incoming data indirect via slave instances. The slave instances behave like a normal instance except they forward all received measurement messages to the master instance.

*2) PushNotificationService*

The PushNotificationService (PNS) component's job is to provide a one-way communication channel to notify participants based on their position, thus safety authorities are able to send information to participants in real time. To achieve the instant messaging functionality, we have to maintain a persistent TCP connection, so if the organizers want to notify all participants in a given cell (as there is a possibility to narrow the notifications to only one cell) they can do it instantaneously. Naturally, PNS is able to store different notification messages for different cells with an expiration date. Every time when a device changes its cell, PNS will send all

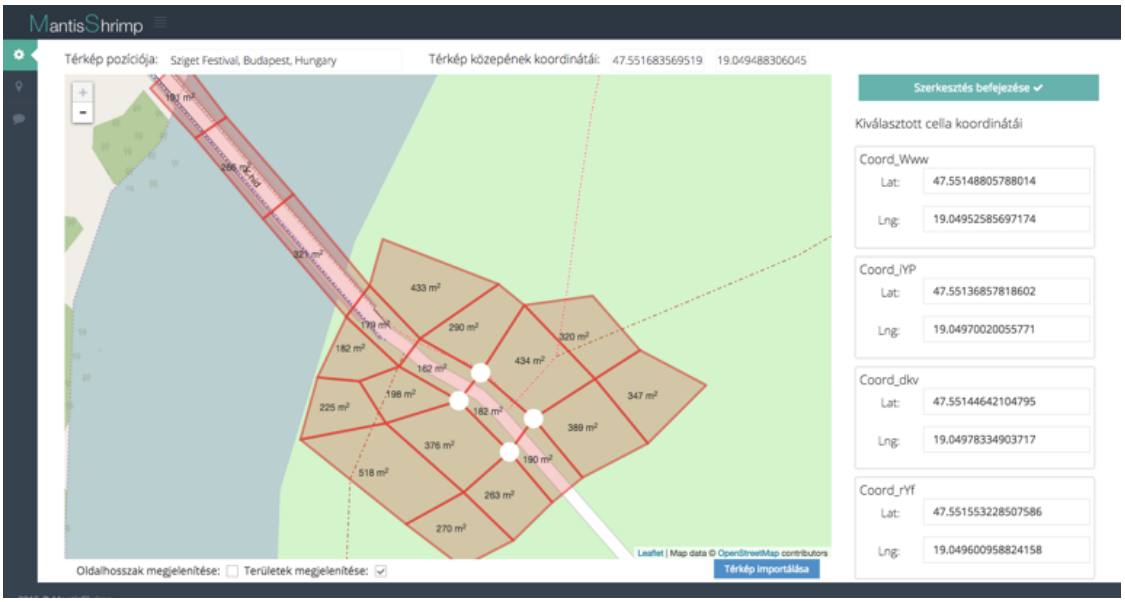


Figure 3. Web based user interface, map editor

notification messages which have been set to the given cell, because the PNS detects the cell handover by analyzing the position message stream from the TAS.

As the PNS needs to maintain lot of persistent connections for the mobile applications, we had to designed it to be able to handle huge number of persistent communication channels. As Figure 2. shows, the component has two types of subcomponents. First is the PushNotificationServer (PS), which is responsible to maintain the communication channels (here we also use Netty) and to store the notification messages. In some cases, one PS can not manage the overall load, thus our service can run on more instances, however then the operation of different instances should be synchronized. For this purpose, we developed a central control element, the PushNotificationController (PNC). On the one hand, PNC maintains the connection with the TAS and distributes the received position messages between the PS instances. On the other hand, PNC also handles and distributes new notification messages received from the safety authorities.

Another important functionality for that we use PNS, is to distribute the current map (current cell division) to the mobile applications (as mentioned earlier), so based on the actual downloaded map they can calculate their actual cell id. With this approach, we can dynamically change the map the organizers require it.

*3) SparkEvaluationService*

The SparkEvaluationService (SES) is responsible for the efficient evaluation of measurement messages. Inside the component, we use the Apache Spark which is a fast and general-purpose cluster computing system. It supports a rich set of higher-level tools including Spark SQL for SQL and structured data processing, MLlib for machine learning, GraphX for graph processing, and Spark Streaming [23] which is the key feature in our case. Spark Streaming in an extension of the core Spark API that enables us to process live data stream from the TrafficAggregatorService. Therefore we have implemented the communication interfaces to the TAS, our evaluation algorithm and our multithreaded thread-safe database access interface to save the evaluation results.

Due to the operation of Spark Streaming our evaluation algorithm runs periodically (period time is so called batch interval) on the actually received measurements so essentially every evaluation result is the change of the crowd's state from the previous crowd's state. Because of this we keep the actual state in the memory for efficient state computing and we always add the changes to it which is really fast in case of heavy loads too. At the end of an evaluation period we save the actual state to the database via our access interface. The SES calculates the following metrics for each cell which have been previously explained in the description of mass surveillance model:

- headcount
- average velocity variance
- crowd density
- crowd pressure

In case of failure, SES is able to recover the last valid status from the database, thus it can continue where it was stopped.

*D. User interface*

When designing the user interface our goal was to build a well useable, simple and logical surface for organizers and safety authorities, to enable them to quickly understand the crowd dynamics tendencies at the different areas of the event. Another important aspect was to be compatible with different resolutions, therefore the choice fell on a web interface.

The main element of the GUI is a map which always contains up-to-date information about the crowd. Similarly to Lord's Mayor Show 2011's system we use different colors to display density and crowd pressure, but instead of heat map we visualize cells. With play button a user can track how the movement of the crowd changed in various time slots. Of course the user can pause or search if he wishes. If he clicks a cell on the map, all historical data will be displayed in a popup window.

We have also implemented a map editor interface (shown on Figure 3.), where the actual map and cell structure can be edited easily, by dragging and moving the coordinates or cells, remove or add new cells, etc. On the left side of the surface, a user can edit the selected cell's exact GPS coordinates. At the end of the editing all changes are imported to the database, if the user clicks to the Map import button.

## IV. SIMULATION RESULTS

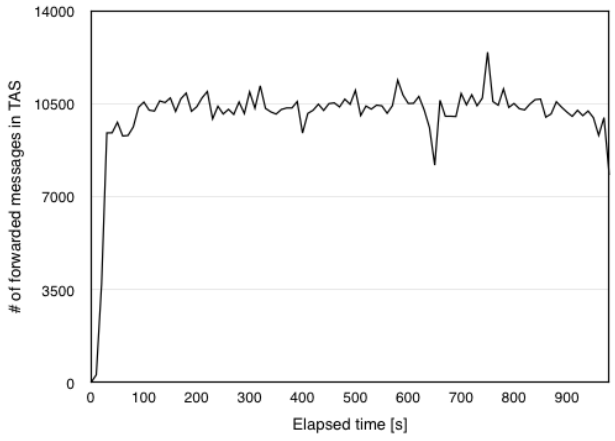


Figure 4. Number of forwarded messages in TrafficAggregatorService

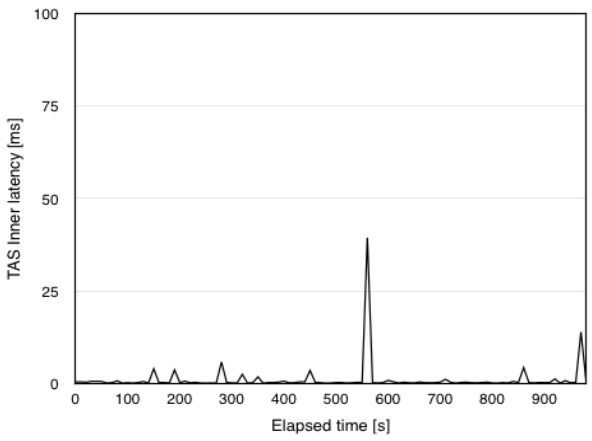


Figure 5. Inner message latency of TrafficAggregatorService

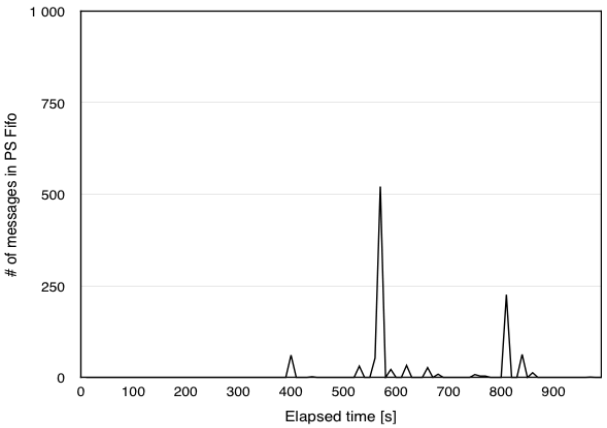


Figure 6. Number of messages in PushNotificationServer message Fifo

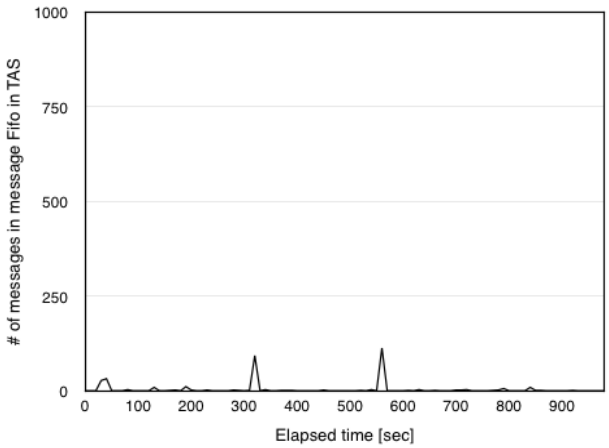


Figure 7. Fill of message Fifo in TrafficAggregatorService

The purpose of the scalability simulations was to validate our system, if it is working correctly and to check what is the maximum load that our solution is able to handle. To perform the tests it was necessary to develop a device simulator application which is able to simulate thousands of devices which are able to communicate properly with the system and use algorithms which simulate the movement of the crowd properly. Unfortunately, it was not possible for us to run tests on a powerful server, thus we ran them on a MacBook Air Mid-2013 (CPU: 1.3GHz i5, Memory: 4GB 1600Mhz DDR3).

Thousands of parallel simulated devices that are connected continuously to the system can not be handled on a synchronous way, as it would need the same number of running threads which would completely exhaust the resources. Therefore, we had to simulate the communication of the devices on an asynchronous way, furthermore we also had to implement the device simulator core using an event-driven paradigm to simulate their functionality properly.

We wanted to apply proven solutions for the crowd movement simulation, therefore the Random Waypoint [24] and the Gauss-Markov [25] mobility model was adapted to our environment.

We started the performance tests with the TrafficAggregatorService. In the case of this component we wanted know how many messages can be transmitted to other components per second, what is the processing time of a message in this component, the operation of the measurement message Fifo inside the component at a given moment and the memory usage of the component. We found that the TAS is able to forward even 10 000 messages per second (Figure 4.). If we assume that a participant changes cell every 5 minutes in average, which can be a valid assumption in a real world operation, it means that the TAS can deal with around 3 million active participants. Meanwhile, we also measured the memory usage of the component and it was not been more than 2,4% in any moment of the testing. We also investigated the saturation of the measurement message Fifo. The load of the Fifo was usually around zero, except for some cases (Figure 7.). The reason for those could be various (GC call, context change, etc.), but the Fifo can store around 4.5 million messages and the measured peak was 110 message thus it is negligible. The average inner message latency of the TAS was 0.42 ms (Figure 5.). Here, there were also some peaks, which were caused by the fact that those stayed more time in the Fifo. The highest latency was 40 ms which is a low value in our case.

We have also investigated the performance of the PushNotificationService. We used the same load (10 000

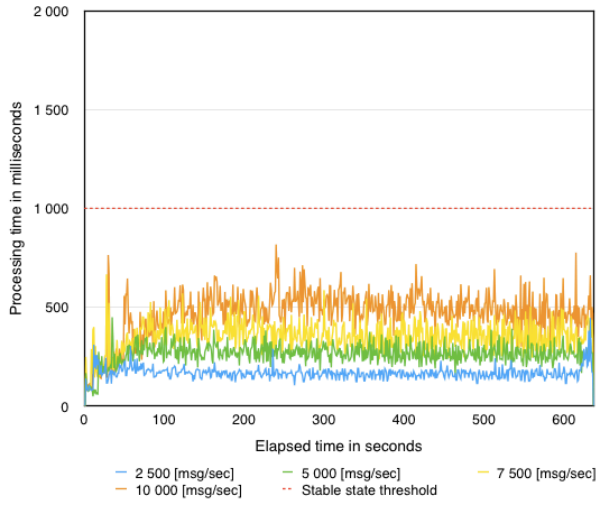


Figure 8. Evaluation's processing time in case of 1 second batch interval

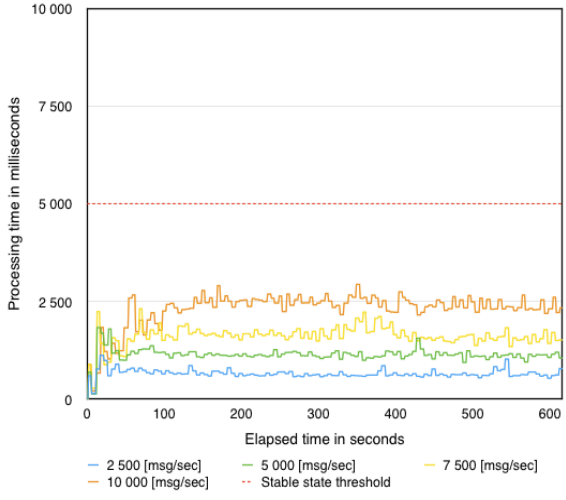


Figure 9. Evaluation's processing time in case of 5 second batch interval

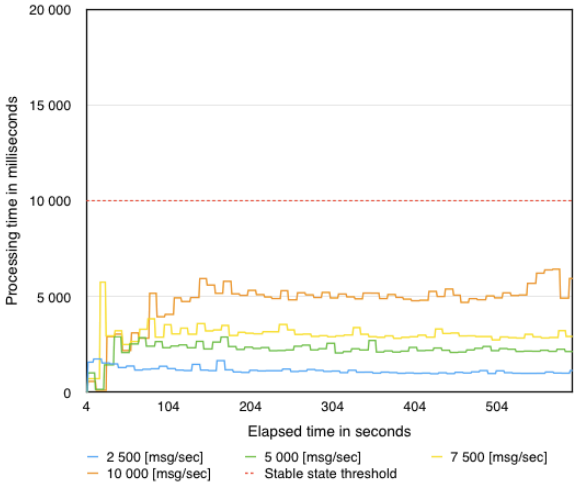


Figure 10. Evaluation's processing time in case of 10 second batch interval

messages per second) for the tests as in the case of the TrafficAggregatorService, using 200 cells and 150 notification messages which can correspond to a real-life setting. We have found similar results as in the case of the TrafficAggregatorService tests. Both components have low memory consumption (under 7%) and in most of the cases the load of Fifos were around zero (Figure 6.). There were also some peaks in the load of the Fifo of the PNS, the highest was 600 messages, it is also negligible compared to the 4.5 million messages.

In our system the Spark Evaluation Service component is responsible for the evaluation of the received measurement messages. Because of its importance, we thoroughly investigated its operation and its performance. We wanted to know what is the maximal load which can be handled by the module, what is the optimal time period (batch interval) to start a new evaluation period or how the measurement message load affects the efficiency of the evaluation.

For a Spark Streaming application operation to be stable, the system should be able to process data as fast as it is being received. In other words, the processing time of the evaluations should be less than the running time of the evaluation algorithm. Another important aspect is that the batch interval should be as low as possible because a lower interval shows more accurate insight into crowd dynamics and it reduces the latency of new crowd states for the organizers.

During the simulations we have also investigated the effect of various input parameters meanwhile we monitored the memory consumption and other Spark specific metrics. From the list of the parameters we would like to highlight the receive rate (or speed) of measurement messages and evaluation algorithm's starting period time (batch interval). batch interval. We have run the simulations with four different receive rates (2500, 5000, 7500, 10000 message/second) and three batch intervals (1, 5, 10 seconds).

During the tests we have measured all 12 possible combinations, the results are shown on Figure 8., 9. and 10. where the curves are grouped by the size of the batch intervals. On the figures the processing time of the evaluation algorithm in Spark can be seen, differentiating among the receive rates by color. On each figure the red dashed lines indicate the maximum processing time for which the component remains stable, being able to handle the measurement message stream with an appropriate speed. The maximum processing time equals with the batch interval size. If the processing time becomes higher than the batch interval, it means that the component is not able to process the received data in the given batch interval, therefore the organizers will be served with obsolete crowd state information. As shown on the figures, even in the case of the highest load (10000 messages/second) the SparkEvaluationService component remained stable. As we increased the rate of incoming measurement messages, the processing time has also increased but the scalability factor stays constant as it scales, which means the system scale well to even an extremely high number of participants. It is also observable on the figures that the size of the batch interval has no effect on this scaling. For example on Figure 8., 9. and 10., the curves for the 10000 message/second receive rate are nearly at the same distance to the red dashed line on all three figures. Based on these observation, we are able use the 1 second batch interval thus we can reduce the latency of new crowd states for the organizers.

## V. Conclusion

Nowadays, the huge mass events have high attendance, therefore the occurrence of dangerous incidents is on the rise.

We have developed a mass surveillance system which provides useful information for the organizers in real time via a simple and logical web interface. The cellular model was modified by us to provide an adequate abstraction which allows us to reduce significantly the number of irrelevant data and the network traffic. In our integrated mass surveillance the different tasks have been distributed among three components so a component failure does not cause the failure of the whole system. Our system was created to be able to receive incoming measurement messages from more than ten or hundred thousands of different sources.

Three crucial components of the implemented system were subjected to detailed simulations tests. The tests showed that all

components are prepared to process and forward even 10 000 messages per second, which means that an event of several hundred thousands of participants could be served by this system. As a next step, we plan to optimize the inner processes to reach an even better and more reliable performance, also adding new functionalities to the user interface.

## REFERENCES


[1] Xiaomeng Shi, Zhirui Ye, Nirajan Shiwakoti, Dounan Tang, Chao Wang, Wei Wang, "Empirical investigation on safety constraints of merging pedestrian crowd through macroscopic and microscopic analysis", Accident Analysis & Prevention, 2015

[2] Á. Petkovics, V. Simon, I. Gódor and B. Böröcz, "Crowdsensing Solutions in Smart Cities towards a Networked Society", Endorsed Transactions of Internet of Things, vol. 15, 2015

[3] Ganti, R. K., Ye, F., and Lei, H., "Mobile crowdsensing: current state and future challenges", Communications Magazine, IEEE, vol. 49, pp. 32-39, 2011

[4] Security NewDesk: How many cameras in the UK? Only 1.85 million, claims ACPO lead on CCTV, online: http://www.securitynewsdesk.com/how-many-cctv-cameras-in-the-uk, accessed: September 2015

[5] Michal Grega, Andrzej Matiolanski, Piotr Guzik and Mikolaj Leszczuk, "Automated Detection of Firearms and Knives in a CCTV Image", Sensors 2016, vol. 16, no. 1,47

[6] Coretta Phillips, A review of CCTV evaluations: "Crime reduction effects and attitudes towards its use", Crime prevention studies,Vol. 1. Criminal Justice Press, Monsey, NY, pp. 123-156., 1999

[7] R. W. Beard, T.W. McLain, D. B. Nelson, D. Kingston, D. Johanson, "Decentralized Cooperative Aerial Surveillance Using Fixed-Wing Miniature UAVs", Proceedings of the IEEE, vol. 94, issue 7, pp. 1306-1324, 2006

[8] Donna Nelson, Traffax Inc., Riverdale MD: Proximity Information Resources for Special Event, 2012

[9] Traffax Inc.: BluFAX Concept, online: http://www.traffaxinc.com/content/blufax-concept, November 2013

[10] Eve Mitleton-Kelly, "Co-evolution of Intelligent Socio-technical Systems", pp.61-77, 2013

[11] Kim, D.H., Kim, Y., Estrin, D., Srivastava, M.B., "Sensloc: Sensing everyday places and paths using less energy" Proceedings of the 8th ACM Conference on Embedded Networked Sensor Systems, Zurich, pp. 43-56, 2010

[12] Dirk Helbing, "A Fluid Dynamic Model for the Movement of Pedestrians", Complex Systems 6, pp. 391-415, 1992

[13] Henderson, L. F., "The Statistics of Crowd Fluids", Nature vol. 229, pp. 381-383, 1971

[14] Victor J. Blue, Jeffrey L. Adler, "Cellular automata microsimulation for modeling bi-directional pedestrian walkways", Transportation Research Part B: Methodological, vol. 35, issue 3, pp. 293-312, 2001

[15] Minoru Fukui and Yoshihiro Ishibashi, "Jamming Transition In Cellular Automaton Models for Pedestrians on Passageway", Journal of the Physical Society of Japan, vol. 68, no. 11, pp. 3738-3739, 1999

[16] Ansgar Kirchner, Andreas Schadschneider, "Simulation of evacuation processes using a bionics-inspired cellular automaton model for pedestrian dynamics", Physica A: Statistical Mechanics and its Applications, vol. 312, issues 1-2, pp. 260-276, 2002

[17] Andreas Schadschneider, "Cellular Automaton Approach to Pedestrian Dynamics – Theory", Pedestrian and Evacuation Dynamics, Springer, pp. 75-85, 2001

[18] P.G. Gipps and B. Marksjö, "A micro-simulation model for pedestrian flows", Mathematics and Computers in Simulation, vol. 27, issues 2-3, pp. 95-105, 1985

[19] Helbing, D., Johansson, A., Al-Abideen, H.Z.: Dynamics of crowd disasters: an empirical study. Phys. Rev. E 75(4), 046109, 2007

[20] Wenjian Yu and Anders Johansson, "Modeling crowd turbulence by many particle simulations", Phys. Rev. E 76, 046105, 2007

[21] M. Zaharia, T. Das, H. Li, T. Hunter, S. Shenker, I. Stoica, "Discretized Streams: Fault-Tolerant Streaming Computation at Scale", Proceedings of the Twenty-Fourth ACM Symposium on Operating Systems Principles, Farmington, pp. 423-438, 2013

[22] Netty project, online: http://netty.io/, accessed: 3/20/2016

[23] Spark Streaming, online: http://spark.apache.org/streaming/, accessed: 3/20/2016

[24] David B. Johnson, David A. Maltz, "Dynamic Source Routing in Ad Hoc Wireless Networks", The Kluwer International Series in Engineering and Computer Science, vol. 353, pp. 153-181, 1996

[25] Liang and Z. Haas, "Predictive Distance-based MobilityManagement for PCS Networks", IEEE INFOCOM 1999, vol. 3, pp. 1377-1384, 1999

[26] A. Lin, J. Zhang, K. Lu and W. Zhang, "An efficient outdoor localization method for smartphones", Computer Communication and Networks, 2014 23rd International Conference, pp. 1-8, 2014

[27] X. Zhu, Q. Li, G. Chen, "APT: Accurate outdoor pedestrian tracking with smartphones", INFOCOM, 2013 Proceedings IEEE, pp. 2508-2516, 2013